```
import seaborn as sns
```

```
df=sns.load_dataset('iris')
```

```
df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
df['species'].value_counts()
```

species	count
setosa	50
versicolor	50
virginica	50

dtype: int64

- You can see that there are three classes

1. setosa
2. versicolor
3. virginica

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
```

```
df['species']=le.fit_transform(df['species'])
```

```
df['species'].head()
```

	species
0	0
1	0
2	0
3	0
4	0

dtype: int64

```
new_df=df[['sepal_length', 'sepal_width',   'petal_length', 'petal_width', 'species']]
```

- I reduce the column of the dataset as I want to show you the graph of the dataset.

```
new_df.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
x=new_df.iloc[:,:-1]
y=new_df.iloc[:,-1]
print(x)
print(y)
```

```
     sepal_length  sepal_width  petal_length  petal_width
0             5.1          3.5           1.4          0.2
1             4.9          3.0           1.4          0.2
2             4.7          3.2           1.3          0.2
3             4.6          3.1           1.5          0.2
4             5.0          3.6           1.4          0.2
..            ...          ...           ...          ...
145           6.7          3.0           5.2          2.3
146           6.3          2.5           5.0          1.9
147           6.5          3.0           5.2          2.0
148           6.2          3.4           5.4          2.3
149           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
```

```
147    2
148    2
149    2
Name: species, Length: 150, dtype: int64
```

```
import pandas as pd
```

```
#x=x.to_numpy()
```

```
#y=y.to_numpy()
#print(y)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=3)
```

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(120, 4)
(120,)
(30, 4)
(30,)
```

```
from sklearn.linear_model import LogisticRegression
```

- **You think why am I import Logistic Regression class??**
- **So the answer is so simple, there is no separate class for apply Softmax Regression in sci-kit learn, there is one perameter named multi_class which is set to 'multinomial' to do Softmax Regression.**

```
smr=LogisticRegression(multi_class='multinomial')
```

it is that much simple to implement

```
smr.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1247: FutureWarnir
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:469: ConvergenceWa
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
▾          LogisticRegression          i ?
LogisticRegression(multi_class='multinomial')
```

```
y_predict=smr.predict(x_test)
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_predict)
```

```
1.0
```

```
print("query= ",x_test.iloc[0])
print("acrual ans= ",y_test.iloc[0])
```

```
query=  sepal_length    4.6
sepal_width     3.2
petal_length    1.4
petal_width     0.2
Name: 47, dtype: float64
acrual ans=  0
```

```
x_test.iloc[0]
```

	47
sepal_length	4.6
sepal_width	3.2
petal_length	1.4
petal_width	0.2

dtype: float64

```
# prompt: do prediction on [4.6, 3.2, 1.4, 0.2] by giving this value I got the prediction.

import pandas as pd
# Create a DataFrame with the input values
new_data = pd.DataFrame({'sepal_length': [4.6], 'sepal_width': [3.2], 'petal_length': [1.4], 'pe

# Make the prediction using the trained model
prediction = smr.predict(new_data)

# Print the prediction
```

```
print("Prediction for [4.6, 3.2, 1.4, 0.2]:", prediction[0])
```

```
Prediction for [4.6, 3.2, 1.4, 0.2]: 0
```

```
if prediction[0]==0:
  print("setosa")
elif prediction[0]==1:
  print("versicolor")
else:
  print("virginica")
```

```
setosa
```